Knowledge and Information Systems

Expert-driven Trace Clustering with Instance-level Constraints

--Manuscript Draft--

Manuscript Number:	KAIS-D-18-00500R2
Full Title:	Expert-driven Trace Clustering with Instance-level Constraints
Article Type:	Regular Paper
Keywords:	Trace clustering; Process Mining; Semi-supervised learning; Constrained clustering
Corresponding Author:	Pieter De Koninck, M.S. KU Leuven Leuven, BELGIUM
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	KU Leuven
Corresponding Author's Secondary Institution:	
First Author:	Pieter De Koninck, PhD
First Author Secondary Information:	
Order of Authors:	Pieter De Koninck, PhD
	Klaas Nelissen, PhD
	Seppe vanden Broucke, PhD
	Bart Baesens, PhD
	Monique Snoeck, PhD
	Jochen De Weerdt, PhD
Order of Authors Secondary Information:	
Funding Information:	H2020 Marie Skłodowska-Curie Actions (822214) — Dr. Jochen De Weerdt
Abstract:	Within the field of process mining, several different trace clustering approaches exist for partitioning traces or process instances into similar groups. Typically, this partitioning is based on certain patterns or similarity between the traces, or driven by the discovery of a process model for each cluster. The main drawback of these techniques, however, is that their solutions are usually hard to evaluate or justify by domain experts. In this paper, we present two constrained trace clustering techniques that are capable to leverage expert knowledge in the form of instance-level constraints. In an extensive experimental evaluation using two real-life datasets, we show that our novel techniques are indeed capable of producing clustering solutions that are more justifiable without a substantial negative impact on their quality.

OUR REFERENCE

YOUR REFERENCE KAIS-D-18-00500R1

LEUVEN

Revision of the manuscript Number KAIS-D-18-00500R2 - "Expert-driven Trace Clustering with Instance-Level Constraints**"**

Dear Prof. Wu,

We would like to explicitly thank you for taking the necessary action to provide us with this conditional acceptance decision. Hereby, we would like to submit our revised manuscript entitled "Expert-driven Trace Clustering with Instance-Level Constraints", to Knowledge and Information Systems.

First of all, we would like to thank the reviewers once again for their input throughout the reviewing process.

We have addressed the remaining remarks from reviewer 7 as follows:

- *"page 8, line 15 : "non-process model aware constrained techniques", but on the same page it is stated that the algorithm mines a process model."*
  - We have clarified this sentence. Our approach is indeed model-driven, but in the section above we explain that we also implemented the expert-driven techniques which are not process model-driven. This is what we refer to.

- *"page 10, line 56: "wel" -> "well" "*
  - We corrected this typo

- *"The comparisons between algorithms is only in terms of quality. Why there is no experimental performance comparison? I find it important to compare performance as well as results quality."*
  - In line with previous comments during the reviewing process, we have analyzed in detail the theoretical time complexity of our algorithm (see Section 3.5). This shows that our technique will run quadratically in the number of clusters desired, and given the complexity of the process model discovery and metric evaluation, polynomial in the number of unique traces and clusters. Albeit interesting, we have not included a detailed experimental performance analysis because it was first of all not a key goal of the technique as presented in this work to make it as efficient as possible. In addition, a decent experimental performance analysis would immediately extend the paper drastically because our own technique and competitors offer for the selection and tuning of several hyperparameters and subroutines (such as process model discovery and process model quality assessment). The theoretical time complexity analysis is in our opinion very detailed and allowing the reader to understand the performance demands of our technique.

Yours sincerely,

Jochen De Weerdt (in name of all co-authors)

JOCHEN DE WEERDT, ASSOCIATE PROFESSOR
TEL. + 32 16 32 88 39
Jochen.deweerdt@kuleuven.be
http://feb.kuleuven.be/research/leuven/LIRIS/

Knowledge and Information Systems manuscript No.
(will be inserted by the editor)

Expert-driven Trace Clustering with Instance-level Constraints

Pieter De Koninck · Klaas Nelissen · Seppe vanden Broucke · Bart Baesens · Monique Snoeck · Jochen De Weerdt

Received: date / Accepted: date

Abstract Within the field of process mining, several different trace clustering approaches exist for partitioning traces or process instances into similar groups. Typically, this partitioning is based on certain patterns or similarity between the traces, or driven by the discovery of a process model for each cluster. The main drawback of these techniques, however, is that their solutions are usually hard to evaluate or justify by domain experts. In this paper, we present two constrained trace clustering techniques that are capable to leverage expert knowledge in the form of instance-level constraints. In an extensive experimental evaluation using two real-life datasets, we show that our novel techniques are indeed capable of producing clustering solutions that are more justifiable without a substantial negative impact on their quality.

Keywords Trace clustering · Process mining · Semi-supervised learning · Constrained clustering

Mathematics Subject Classification (2000) 62H30 · 91C20

1 Introduction

Process mining is a research field at the crossroads of data mining and business process management. Its main reason for existence stems from the vast amount of data that is generated in modern information systems, and the desire of organizations to extract meaningful insights from this data. Generally speaking, three subdomains exist within process mining: process discovery, a set of techniques concerned with the elicitation of process models from event data; conformance checking, a set of techniques that aim to quantify the conformance between a certain process model and a certain event log; and process enhancement, approaches that aim to

P. De Koninck, K. Nelissen, S. vanden Broucke, B. Baesens, M. Snoeck, J. De Weerdt
KU Leuven, Research Center for Management Informatics (LIRIS), Naamsestraat 69, B-3000 Leuven, Belgium E-mail: pieter.dekoninck@kuleuven.be
B. Baesens
University of Southampton, Southampton Business School, Southampton, United Kingdom

extend existing or discovered process models by using other data attributes such as resource or timing information [30].

One of the main challenges in applying techniques from the process discovery subdomain to real-life cases, however, is that the event data corresponding to these cases typically contain highly varied and complex behavioural structures. This leads to a lower quality of the process models which can be discovered. Multiple avenues for mitigating this issue have been proposed, such as focusing on models that hold locally rather than globally [29], applying techniques from the sequential pattern mining domain to extract frequent patterns rather than process models [22], improving the abstraction level of the data [23] and partitioning the event log into separate clusters, also called trace clustering [28].

Trace clustering improves the quality of a process discovery exercise by splitting a highly varied event log into several different clusters of traces, and then discovering a process model for each trace cluster separately. This should decrease the variability of behaviour present in each cluster, and lead to a higher quality of the discovered process models. There have been successful applications of trace clustering techniques in a wide variety of contexts, ranging from e.g. incident management to healthcare [14, 15].

Nonetheless, trace clustering, like traditional data clustering, is hindered by its unsupervised nature: it is often hard to validate a clustering solution, even for domain experts. This problem has been recognized in [9], in which an approach is proposed to increase understandability of trace clustering solutions by extracting short and accurate explanations as to why a certain trace is included in a certain cluster. Although explaining cluster solutions to domain experts is a valid approach for enhancing the understandability of trace clustering solutions, it remains a post-processing step. A potentially better approach for improving trace clustering solutions is to directly take an expert's opinion into account while performing the clustering.

As such, the core contribution of this paper is the proposal and evaluation of two novel types of trace clustering techniques: similarity-driven (or process model agnostic) and process model aware constrained trace clustering techniques. These techniques incorporate expert knowledge, in the form of instance level must-link and cannot-link constraints, into the trace clustering algorithm. In an experimental evaluation using two real-life datasets, these approaches are shown to lead to clustering solutions that are more in line with the expert's expectations, without substantially diminishing the quality of the clustering solution.

In light of this objective, the rest of this paper is structured as follows: in Section 2, the fields of trace clustering and constrained clustering are described. In Section 3 their strengths are combined, leading to our proposed approaches. Subsequently, the contribution of our novel approaches is evaluated in Section 4. Finally, a conclusion and outlook towards future work are provided in Section 5.

2 Related Work

In this section, first, a short overview of trace clustering is provided. Then, the concepts of the constrained clustering fields are described, since they provide an interesting avenue for incorporating expert knowledge. Finally, we combine both aspects to assess the research gap that exists regarding constrained trace

clustering: no trace clustering technique has been proposed that can take into account expert knowledge, except for [10], in which a full expert-based clustering solution is required. As it is quite unrealistic to expect that experts can give such a complete clustering, this paper proposes two distinct approaches to take into account expert information in the form of constraints while clustering traces.

2.1 Classical Trace Clustering

Typically, the starting point of a trace clustering exercise is an event log, which is a set of traces. Each trace is a registered series of events (instantiations of activities), possibly along with extra information on the event, such as the resource that executed the event or time information. A trace clustering is then a partitioning of an event log into different clusters such that each trace is assigned to a single cluster.

A wide variety of trace clustering techniques exist. Broadly speaking, there are three main categories of trace clustering techniques: those based on direct instance-level similarity, those based on the mapping of traces onto a vector space model, and those based on process model quality. With regards to direct instance-level similarity, i.e. the direct quantification of the similarity between two traces, an adapted Levenshtein distance could be computed as in [4]. An alternative set of approaches are those where the behaviour present in each trace is mapped onto a vector space of features [5, 15]. The third category considers process model quality as an important goal for trace clustering. An approach based on the active incorporation of the process model quality of process models discovered from each cluster has been described in [14]. An older approach based on representing the traces in each clusters with Markov Chains has been proposed in [34].

2.2 Incorporating Expert Knowledge: Constrained Clustering

Constrained clustering typically deals with forcing certain instances to be clustered together (must-link or positive constraints), or in separate clusters (cannot-link or negative constraints) [35]. A wide array of clustering techniques have been extended to incorporate such constraints, including partitional approaches such as k-means clustering [35], hierarchical clustering [7], model-based clustering with Expectation-Maximization [20], spectral approaches [37], and multi-view clustering [17], among others. Observe that other types of constraints have also been proposed: examples include the use of multi-instance constraints [19] or constrained cluster sizes [39]. Constrained clustering has been applied in activity clustering, where the clustering is focused on grouping activities together, rather than partitioning event logs [36].

Specific attention is given to the quality of the constraints, as it has been shown that their inclusion can lower the performance of clustering techniques. Davidson and Ravi [8] define the *informativeness* of a constraint set as the amount of information contained in the set that the algorithm would not be able to infer on its own. If the objective function or bias of the clustering technique is different from the preference of the constraints, the resulting clustering solution will differ significantly from a situation where no constraints were included. Such a constraint

set has a high informativeness. Furthermore, a concept of *coherence* between constraints with regards to a distance function is intuitively described as follows: to have a high coherence, must-link and cannot-link constraints should not contradict each other by connecting multiple points from the same neighbourhood. If points a and b are similar, and so are c and d, then it makes little sense to have a must-link constraint between a and c while also having a cannot-link between b and d.

2.3 Constrained Trace Clustering

Three distinct avenues for the incorporation of expert knowledge in trace clustering are discussed in [10]: expert seeding, constrained clustering and complete expert pre-clustering. The approach that is subsequently developed in [10] is based on the latter category: it requires a completely pre-clustered set of traces to be provided by the expert. In this paper, we extend the technique in [10] by proposing a new algorithm (ConDriTrac) that makes it possible to shift from a full expert solution to a small set of constraints in terms of expert input. Furthermore, adapted versions of process model agnostic techniques presented in [4] and [5] that incorporate constraints are proposed as well, denoted as *Constrained direct* and *Constrained vector-based* techniques below.

	Unsupervised	Expert-Driven
Process Model Agnostic	Hierarchical direct distance: e.g. GED [4] Hierarchical vector-based distance: e.g. MRA, kgram [5] Spectral vector-based [15]	Constrained direct* Constrained vector-based*
Process Model Aware	Expectation-Maximization Markov Model Clustering [34] Active trace clustering [14]	ActSemSup [10] ConDriTraC*

Fig. 1 Classification of trace clustering techniques based on process model awareness and incorporation of expert knowledge. Techniques denoted with a "*" are proposed in this paper.

Figure 1 illustrates the research gap that is being addressed in this paper. From the discussion in Section 2.1, it is clear that one of the dimensions for classifying trace clustering techniques is whether or not they are *process model aware*: is the clustering procedure guided by an underlying mined process model representing each cluster? As an alternative, clustering techniques can also be *process model agnostic* in which case the clusterings are determined based on the intrinsic similarities of the traces, thus without the need of a process model.

Orthogonal to the trace clustering dimension is the aspect of expert supervision. From Figure 1, it can be seen that all but one trace clustering technique cannot take expert knowledge into account. As such, we specifically address this research gap by proposing both process model-aware as well as process model-agnostic trace clustering techniques that can be guided by an expert in the form of cannot-link and must-link constraints.

3 Incorporating Expert Knowledge in Trace Clustering

This section introduces our proposed constrained trace clustering techniques. Before outlining the methods themselves, we formally define the necessary background concepts in order for the reader to understand the algorithmic details further in the text.

3.1 Preliminaries

An event log is a collection of traces (also called process instances). Each trace contains an identifier, a sequence of events, and optional other attributes. Events can be ordered simply as a sequence, but are typically denoted based on a time stamp.

Definition 1 (Event) An event is a tuple $e = (p, \tau, a, x_1, ..., x_n)$ where p is the identifier of the trace it belongs to, τ the timestamp, a the activity label, and $x_1, \ldots, x_n$ any number of additional attributes. Labeling functions $tid : e \mapsto p$, $time : e \mapsto \tau$, and $act : e \mapsto a$ are included.

Definition 2 (Trace) A trace is a finite sequence of events $t = \langle e_1, \ldots, e_{|t|} \rangle$, with $|t|$ the number of events in that trace. The events are sequenced based on their time stamp τ such that $\forall e_i, e_j \in t : i < j \rightarrow time(e_i) \leq time(e_j)$. The trace identifier is contained in its events: $\forall e_i, e_j \in t : tid(e_i) = tid(e_j)$. This identifier can be retrieved through the labelling function $tid : t \mapsto tid(e_1) = p$. Two traces are equal if they have the same identifier: $t = s \Leftrightarrow tid(t) = tid(s)$.

Definition 3 (Event Log) An event log L is a set of traces. $|L|$ denotes its cardinality.

Sometimes, it is more useful to group traces that have the same control-flow pattern (disregarding their trace identifiers or timestamp). We refer to such traces as *distinct process instances*. An event log where traces are identified purely based on their sequence of activity labels is called a grouped event log.

Definition 4 (Grouped Event Log) In a grouped event log G, traces are considered equal if their sequence of events is equal: $t = s \Leftrightarrow |t| = |s| \wedge \forall i = 1, ..., |t| : act(t_i) = act(s_i))$. An element of a grouped event log is called a distinct process instance. The grouped event log G is a multiset of such distinct process instances, with $supp(G)$ denoting the number of distinct traces, or support of the multiset. $|G|$ denotes the cardinality, or its total size. The frequency of a distinct trace is the multiplicity of that trace in the grouped event log.

Definition 5 (Trace Clustering) A trace clustering C is a partition of an event log L: a set of nonempty subsets of L such that the union of all clusters is equal to the event log, and none of the clusters overlap: $\bigcup_{A \in C} A = L \wedge \forall A, B \in C : A \cap B \neq \emptyset \rightarrow A = B$.

Definition 6 (Must-link Constraint) A must-link constraint ml is a relation over an event log L presented as an unordered pair of trace identifiers $\{p, r\}$. A trace clustering C satisfies the constraint if there is a cluster A in C which contains both p and r: $\exists A \in C : p \in A \wedge r \in A$.

Definition 7 (Cannot-link Constraint) A cannot-link constraint cl is a relation over an event log L presented as an unordered pair of trace identifiers $\{p, r\}$. A trace clustering C satisfies the constraint if there is no cluster in C which contains both p and r: $\nexists A \in C : p \in A \wedge r \in A$.

Definition 8 (Constraint set) A constraint set CS over an event log L is the union of ML, a set of must-link constraints, and CL, a set of cannot-link constraints, $CS = ML \cup CL$.

Over a set of constraints, reasoning can be applied to explicitly include all implied constraints. The resulting set is denoted here as the transitive extension. The combination of a must-link and cannot-link constraint is considered to be **transitive**, meaning that if 'a must be linked to b' and 'b cannot be linked to c', then 'a cannot be linked to c' either. The same holds for transitivity over two must-link constraints.

Definition 9 (Transitive extension)

The transitive extensions ML^+ and CL^+ are defined over the constraints sets ML and CL containing all implicit must-link and cannot-link constraints respectively.

To formally define these extensions, let us first introduce a boolean function $P(\{x, y\}, ML)$ indicating whether there exists a path of must-link constraints between x and y. Please observe that a valid expert input would require that: $\forall \{x, y\} \in CL : \neg P(\{x, y\}, ML)$. The set of all extended must-link constraints ML^+ is:

$$ML^+ = \{\{x, y\} | P(\{x, y\}, ML)\}$$

Similarly, CL^+ includes all implicit cannot-link relationships:

$$CL^+ = \{\{x, y\} | \exists a : \{x, a\} \in ML^+ \wedge \{y, a\} \in CL\}$$

Finally, let $CS^+ = ML^+ \cup CL^+$.

Definition 10 (Connected traces) Define the set of all traces which are related to a single trace t, given a certain constraint set CS, including t, as the connected traces for t on CS. It is given by the function $cont : (t, CS) \mapsto cont(t, CS) = \{s \| \{s, t\} \in CS\} \cup t$.

Definition 11 (Process model) A process model PM is a diagrammatic representation of a process.

A multitude of languages exists to model processes using diagrams, ranging from flowcharts and UML activity diagrams to workflow nets and BPMN models. For more information, we refer the interested reader to [16].

Definition 12 (Process discovery technique) A process discovery technique PD is a function that maps an event log L onto a process model PM. $PD : L \mapsto PD(L) = PM$.

Definition 13 (Process model quality metric) A process model quality metric m is a function which returns a numeric value given an event log L and a process model PM. $m : (L, PM) \mapsto m(L, PM)$.

Within the field of process mining, measuring the fit between a process model and an event log is often referred to as conformance checking [27]. A wide range of specific metrics have been proposed in the literature, typically focusing on dimensions such as recall (e.g. [1]) and precision (e.g. [32]).

3.2 Constrained Direct and Vector-based Clustering techniques

In this section, a description is given on how the most prevalent set of trace clustering techniques can be adapted for constraints. A number of approaches rely on the quantification of the similarity between two traces [4, 5]. A first approach is to define similarity or distance between two traces directly: an example is through calculating the number of deletions, insertions, and substitutions it takes to go from trace 1 to trace 2: this what we denote with *GED*, or Generic Edit Distance. The result is a matrix containing the pairwise distances between each of the traces. Applying a standard clustering algorithm, such as Agglomerative Hierarchical Clustering, results in a clustering solution. Similarly, in [5], approaches are described for defining a vector-space by calculating features using all traces in an event log, and then the distance between traces is represented by the distance between their vectors in the model. These features can be simple: one example are k-grams, activities that appear together (a three-gram is then a set of 3 activities that appear together in some of the traces), or more complex: several featurizations are proposed in [5] that aim to incorporate common process model characteristics, such as parallelism. The Maximal Repeat Alphabet Feature Set or *MRA* is taken as an example of such featurization. Once the features are chosen, all traces can be mapped onto the vector space model. Given the vector space model and a method of quantifying the distance between vectors (Euclidian, Manhattan, etc.), a matrix with pairwise distances between traces is obtained. On that matrix, a standard data clustering technique can be applied, such as Agglomerative Hierarchical Clustering, resulting in a clustered event log.

Our proposal for making such a technique constraint-aware is simple: given pairwise constraints as input, the goal is to adapt the matrix with pairwise distances to incorporate these must-link and cannot-link constraints. This idea is in line with [7]. Under the assumption that the matrix contains distances, the solution is to change the pairwise distances to a very small number when two instances have to be linked, and to a large number when they cannot be linked. If the matrix at hand contains similarities rather than distances, these approaches can be flipped: large similarities are induced for must-links and small similarities for cannot-links. These algorithmic concepts have been implemented as a plugin for ProM 6[1], and are publicly available from the package ExpertTraceClustering.

[1] ProM is the leading open-source process mining framework for academicians and practitioners, see: `promtools.org`.

For specific configuration of these numbers, preliminary testing has led to the following setup when performing agglomerative hierarchical clustering with Ward's minimum variance method: the distance between two traces is set to zero when they must be linked, and to the maximum distance present in the original matrix times half the number of traces when they cannot be linked.

3.3 ConDriTraC: Process Model Aware Constraint-driven Trace Clustering

In this section, a novel trace clustering algorithm, *ConDriTraC*, which stands for Constraint-driven Trace Clustering is described. It is designed specifically to be driven by expert knowledge. This algorithm has also been implemented as a plugin for ProM 6, and is publicly available in the same package *ExpertTraceClustering* as the non-process model aware constrained techniques explained above. The technique is based on the approach described in [10], where the expert knowledge had the form of a complete expert clustering. Here, the algorithm is adapted to work with constraints as input instead.

In general, the technique consists of three phases:

Phase 1 An initialization phase, during which the clusters are initialized.
Phase 2 A trace assignment phase, during which traces are assigned to the cluster which leads to the best results, if that best result is sufficiently good.
Phase 3 A resolution phase, during which traces that were not assigned in the previous phase, are either included in an additional separate cluster, or in the best possible existing cluster.

Phase 1: Initialization. The first phase is an initialization phase, as presented in Algorithm 1. In this phase, a k-bounded maximal clique of extended cannot-link constraints is determined. In this case, a clique of cannot-link constraints is a set of traces such that all traces in the set are connected through cannot-link constraints in a pairwise fashion. Such a clique is maximal if no trace can be added without the loss of the clique property. Contrary to finding a largest maximal clique, the search is bounded by the number of clusters k so that this procedure can be stopped once a clique is found with a size equalling k. If no such clique can be found, the search will return the largest maximal clique (with its size then being lower than k). Next, each member of the obtained clique is included in a separate cluster, along with all traces that must be linked to it. At that point, if there are less clusters than requested, initialize the remaining clusters randomly from all remaining traces, without violating cannot-link constraints.

Phase 2: Trace Assignment. After the initialization, the set of remaining traces to be clustered will be assigned to the cluster they fit best with. This is done by mining a process model, and calculating the *trace metric value* and *cluster metric value* for each cluster. The *cluster metric value* is based on the correspondence of all traces to the model, both those that were previously added to the cluster and the ones currently being tested. The *trace metric value*, on the other hand, corresponds to the result the metric returns based on only the traces that will be added to the mined process model. For each trace t, the union of t and the traces it must be linked to is denoted as $cont(t, ML)$, see Definition 10. If there exists a cannot-link constraint between at least one of the traces of a cluster, and any trace in $cont(t, ML^+)$, the cluster does not need to be checked. Four situations are

Algorithm 1 ConDriTraC - Constraint-driven Trace Clustering

Input: G := Grouped Event Log, k := the desired number of clusters, CS := a constraint set, cvt := cluster value threshold, tvt := trace value threshold; $SeparateBoolean$:= true if unassignable traces should be grouped in a separate cluster;
Input: Configuration: PD:= a process discovery technique, m:= a process model quality metric
Output: C := An ordered set of clusters

Phase 1: Initialization

1: C:=$\emptyset$
2: L:=G % L is set of unclustered traces
3: Obtain a k-bounded maximal clique in CL^+.
4: For each trace s in this clique, add a cluster to C and add $cont(s, ML^+)$ to it. L:= $L \setminus cont(s, ML^+)$.
5: If $|C|< k$, choose random trace s from L and add $cont(s, ML^+)$ to C as new cluster until $|C|= k$. Remove $cont(s, ML^+)$ from L.

Phase 2: Trace assignment

6: $U := \{\}$ % List of unassignable traces
7: Order L by multiplicity
8: **for** $t \in L$ **do** % Loop over the distinct traces which were not assigned to a cluster in Phase 1
9: $bestCluster := -1$ % Temporary value for assignment
10: $bestCMV := -1; bestTMV := -1$; % Temporary values for optimization
11: **for** $c := (0 \rightarrow |C| - 1)$ **do** % Inspect each possible cluster
12: **if** $\nexists\ s \in C_c$: $s, t \in CL^+$) **then** % There must not be a cannot-link relation between t and any of the traces already in this cluster
13: $PM := PD(C_c \cup cont(t, ML^+))$ % Mine a process model including traces $cont(s, PM^+)$
14: $tmv := m(PM, cont(t, PM^+))$ % Get result of metric on just these traces
15: $cmv := m(PM, C_c \cup cont(t, PM^+))$ % Get result of metric on all traces in cluster c
16: **if** $(tmv >= tvt) \wedge (cmv >= cvt)$ **then** % Check thresholds
17: **if** $cmv > bestCMV \vee (cmv = bestCMV \wedge tmv > bestTMV)$ **then**
18: $bestCMV := cmv; bestTMV := tmv; bestCluster := c$
19: **end if**
20: **end if**
21: **end if**
22: **end for**
23: **if** $bestCluster >= 0$ **then** % If the traces $cont(t, ML^+)$ could be assigned to a cluster
24: $C_{bestCluster} := C_{bestCluster} \cup cont(t, ML^+)$ % Add trace to cluster
25: **else**% If the traces $cont(t, ML)$ could not be assigned to a cluster
26: $U := U \cup cont(t, ML^+)$ % Add traces to unassignable
27: **end if**
28: $L := L \setminus cont(t, ML^+)$ % Remove traces from log
29: **end for**

Phase 3: Unassignable resolution

30: **if** $SeparateBoolean$ **then**
31: $C_{n_b+1} := U$ % Add remaining traces to a new cluster
32: **else**
33: Add each trace to the cluster in CS using the same procedure as in phase 2, without checking the thresholds anymore (drop requirements in line 20). Furthermore, the trace and cluster metric values are now calculated without rediscovering a process model each time.
34: **end if**
35: **return** C

possible: (1) the *cluster metric value* is the highest one, in which case the cluster is denoted as the current best; (2) the *cluster metric value* is only equal to the current highest value but the *trace metric value* is higher than the current best, in which case the cluster is also denoted as the current best; (3) the values are above the threshold but lower than the current best found in one of the other clusters, in which case the trace will not be added to the cluster which is currently being tested; or (4) these values are below the provided thresholds, and again the trace will not be added to the cluster which is currently being tested.

After determining the best cluster, the set of instances are added to the best possible cluster. If no best possible cluster exists (because the metric values were below the threshold for each of the clusters), the instances are added to the set of unassignable traces.

Phase 3: Unassignable resolution. In the third phase, any remaining traces which were not assigned to a cluster in Phase 2 will be assigned to a cluster. They are either added to a separate cluster (if *SeparateBoolean* is true), or they are added to the best possible existing cluster. Assigning them to a separate cluster creates a sort of 'surplus'-cluster, and should make the process models corresponding with the normal clusters of higher quality. However, observe that cannot-link violations within this surplus-cluster are possible. Therefore, in most cases, it makes more sense not to create this extra cluster. Assigning the traces to the best existing cluster, is done following the same procedure as in Phase 2, with the two exceptions: on the one hand, the trace value metric and cluster value metric are calculated compared to the process models obtained for each cluster after Phase 2, and no longer rediscovered before each assignment (line 17 is disregarded). On the other hand the thresholds no longer need to be checked (line 20 is disregarded). Because the maximal clique of cannot-links was used to initialize the clustering in Phase 1, there is always at least one cluster to which each trace can be assigned without violating the cannot-link and must-link constraints.

3.4 Configuration of *ConDriTraC*

In this subsection, a small discussion is provided on how *ConDriTraC* could be configured. While the choice of the two thresholds and the choice whether or not to separate the not-assignable traces are important decisions, these are case-specific decisions. The algorithm allows for parameter configuration: higher thresholds combined with the separation of traces that do not exceed these thresholds will likely lead to small but high quality clusters and one large surplus-cluster, which may be desirable in some cases but not always.

In terms of the metric chosen as input for the clustering, this depends on the expectation of the underlying process models. A wide array of accuracy and simplicity metrics for discovered process models have been described in the literature (e.g. [13]). In general, a weighted metric such as the robust F-score proposed in [12] might be appropriate, since it provides a balance between fitness and precision.

A similar argument holds for the process discovery technique one could use. A wide array of techniques exist, and our approach can be combined with most of them. Observe that the chosen technique should be able to discover processes with a decent scalability, since a high number of process models needs to be discovered in certain steps of our algorithm. Whereas in [14], the preference goes to Heuristics Miner [38], *ConDriTraC* relies on the more recent and robust Fodina process discovery technique [33]. Alternative options include well performing algorithms like Inductive Miner [21] or Split miner [2]. Note also that, currently, the focus is on procedural discovery techniques, though an extension towards declarative techniques or mixed-model paradigms [11] could be considered in future work as well.

3.5 Theoretical complexity

The theoretical complexity of the Constrained Direct techniques and Constrained Vector-based techniques is dependent on two steps: first, the calculation of the pairwise similarity (for the direct technique), and the calculation of the features present in the vector space (for the vector-based techniques). The former has quadratic time complexity, while the latter has a linear time complexity [5]. The second step is applying agglomerative hierarchical clustering, standard complexity of which has been shown to be at least $\mathcal{O}(n^2)$ [26]. In this specific context, n is the number of unique traces. Combining the complexity of both steps, we can conclude that the Constrained Direct techniques and Constrained Vector-based techniques will have a quadratic time complexity.

The theoretical complexity of ConDriTraC, on the other hand, is largely dependent on the complexity of the chosen process discovery technique, which have been shown to be non-linear in the number of activities present in the event log [33]. Furthermore, the calculation of the evaluation metric, and the clustering structure play an important role as well.

Algorithm 1 lists three distinct phases: initialization of the clusters, trace assignment, and resolution of unassignable traces. With regards to the initialization phase, let c be the number of traces included in at least one cannot-link constraint, m the number of cannot-link constraints, and k the desired number of clusters. The goal is then to obtain a clique with a size equal to k. As shown in [6], this can be done in $\mathcal{O}(k \times c^k \times k^2) = \mathcal{O}(c^k \times k^3)$ (lines 1-5). For the second phase, with n referring to the number of unique traces in the event log, and k again the requested number of clusters, each step will discover at most k process models, and calculate at most $2k$ process metrics (lines 11-22). This block is thus completed in $\mathcal{O}(k(PD + ME))$, where $\mathcal{O}(PD)$ and $\mathcal{O}(ME)$ are the complexities of the discovery techniques and the metric evaluation, respectively. This is repeated n times (lines 8-29), resulting in an overall complexity for the second phase of $\mathcal{O}(nk(PD + ME))$. The final phase reassigns the traces that did not meet the thresholds again (lines 30-34), now without rediscovering a process model each time, so worst case, it will have a complexity of $\mathcal{O}(nk(ME))$.

To summarize, the algorithm will run in $\mathcal{O}(c^k \times k^3) + \mathcal{O}(nk(PD + ME)) + \mathcal{O}(nk(ME))$, i.e. quadratic in the number k of clusters desired, and given the complexity of the process model discovery and metric evaluation, polynomial in the number of unique traces n and clusters k.

4 Experimental evaluation

In this section, we will apply a number of existing trace clustering techniques, and several expert-driven trace clustering techniques, on two datasets. The main objective of the experimental evaluation is to demonstrate that our newly proposed constrained trace clustering techniques are indeed capable of producing more justifiable clustering solutions while at the same time not deteriorating clustering quality. Therefore, the obtained clustering solutions are compared in terms of process model quality and justifiability.

4.1 Setup

Data sets. The first dataset, MUNICIP, is a set for which the ground truth is known. This event log is a pre-processed version of the dataset used in the BPI Challenge of 2015, a collection of event data from the permit processes of five Dutch municipalities [31]. The log contains 5649 traces, of which 2502 are distinct. The dataset contains 29 types of activities. A number of additional statistics is presented in Table 1. The five distinct municipalities are considered to be the "true" clusters of traces.

The second data set, TABREAD, is described in [10], and contains logged behaviour of participants in a study regarding tablet newspaper reading. The ground truth here stems from a description of cluster structures by a marketing expert, based on information such as the reading moment, length of a session, how focused a reader is, how thoroughly the paper is read, etc. The expert knowledge is not expected to correlate with a trace clustering view over the data. The created event log contains a wide variety of behaviour: out of 2900 reading sessions, there are 2794 distinct variants. This log contains 34 activity types. A number of additional statistics is presented in Table 1. Furthermore, a small example is presented in Table 2.

Table 1 Data set statistics

Data set	Traces	Distinct traces	Act. types	Average trace length	Min trace length	Max trace length	Average Act. types per trace
MUNICIP	5649	2502	29	16.4	3	76	9.5
TABREAD	2900	2794	31	20.4	2	115	11.1

Table 2 Example event log of the tablet reading process

Session	Time	Activity Type	User
1	16-06-2015 08:02	launch	John Doe
1	16-06-2015 08:03	read-page-front	John Doe
1	16-06-2015 08:03	read-page-politics	John Doe
1	16-06-2015 08:04	scan-page-politics	John Doe
1	16-06-2015 08:04	inspect-image-sport	John Doe
1	...	...	...
1	16-06-2015 08:24	quit	John Doe
2	16-06-2015 08:32	launch	Jane Doe
2	...	...	...

Expert knowledge. The expert knowledge is captured in constraints, which are generated randomly from the ground truth for both data sets, with an equal distribution of must-link and cannot-link constraints. Three sets of constraints are included: 1%, 5%, and 10%, where each set contains a number of constraints equal to said percentage of the number of distinct process instances. The smaller constraint sets are subsets of the larger constraint sets.

Table 3 Clustering techniques compared in the experimental evaluation

Shorthand	Technique	Implementation (Plugin/package)	Process model aware	Expert driven
GED	AHC - Generic Edit Distance	GuideTreeMiner (ProM 6)		
MRA	AHC - Maximal Repeat Alphabet	GuideTreeMiner (ProM 6)		
3-gram	AHC - 3-grams	GuideTreeMiner (ProM 6)		
ActFreq	Frequency-based ActiTraC	ActiTraC (ProM 6)	✓	
ActMRA	Distance-based ActiTraC	ActiTraC (ProM 6)	✓	
ConGED	Constraint-based AHC	own plugin (ProM 6)		✓
ConMRA	Constraint-based AHC	own plugin (ProM 6)		✓
Con3-gram	Constraint-based AHC	own plugin (ProM 6)		✓
ConDriTraC	Constraint-based ActiTraC	own plugin (ProM 6)	✓	✓

AHC: Agglomerative Hierarchical Clustering
Configuration of *ConDriTraC*: *PD*:= Fodina, *m*:= F1-Score, *SeparateBoolean*:= False
For TABREAD: *cvt*:=0.27, *tvt*:=0.27; for MUNICIP: *cvt*:=0.50, *tvt*:=0.25;

Techniques. All included techniques are listed in Table 3, with an indication of whether they are model aware or not and expert-driven or not. Five classical trace clustering techniques are incorporated for comparison: *ActFreq* and *ActMRA* [14], two process model aware techniques, *GED* [4], a direct instance-similarity technique, and two vector-space model-based methods, *MRA* [5] and *3-gram* [28]. Three clustering approaches are included that take expert knowledge into account but are not model aware: the constrained direct technique *ConGED* and two constrained vector-based, i.e. *ConMRA* and *Con3-gram*. Finally, one novel expert-driven trace clustering approach which is process model aware is included: *ConDriTraC*.

Metrics. To evaluate the quality of the clustering solutions, a process model is mined for each cluster, using the Fodina technique [33]. The accuracy of each process model discovered per cluster is then measured using the F1-score as proposed in [12], where p_B is a precision metric and r_B is a recall metric:

$$F1_B = 2 * \frac{p_B * r_B}{p_B + r_B}$$

In this paper, the recall metric we have chosen is behavioural recall $\mathrm{r_b}$ [18], and the precision metric we use is $\mathrm{etc^p}$ [25]. Finally, a weighted average F-score metric for the entire clustering solution is then calculated as follows, similar to the approach in [14], where k is the number of clusters in C and n_i the number of traces in cluster i:

$$F1_C^{WA} = \frac{\sum_{i=1}^{k} n_i F1_i}{\sum_{i=1}^{k} n_i}$$

Furthermore, we can calculate the relative improvement of a technique with expert knowledge (EK) compared to the best pure trace clustering technique (TC) as follows:

$$RI(EK, TC) = \frac{F1_{EK}^{WA}}{F1_{TC}^{WA}}$$

Three situations might arise: (1) $RI > 1$: in that case, the expert-driven technique creates a solution which is able to combine higher ease-of-interpretation with better results in terms of process model quality; (2) $RI = 1$: the expert-driven technique leads to higher ease-of-interpretation from an expert's point of view

without reducing model quality; and (3) $RI < 1$: there is a trade-off present between clustering solutions which are justifiable for an expert and the optimal solution in terms of process model quality.

Finally, we propose measuring the justifiability of a solution. As defined in [24] for classification, justifiability measures the extent to which the results of a classification exercise is in line with existing domain knowledge. Here, justifiability can be measured as the extent to which an expert's expectations are fulfilled: on the one hand, by calculating the percentage of constraints that are violated by a trace clustering solution. These results are calculated based on the non-propagated versions of the constraint set, to avoid calculating the same violation multiple times. As described in [8], the utility of a constraint set can be measured by its informativeness: the extent to which the constraints add information an unconstrained algorithm is not able to infer on its own. If unconstrained approaches violate a high number of constraints, this can be interpreted as high informativeness. On the other hand, we can use clustering indices to compare how similar two clustering solutions are. Ideally, a clustering solution should be closely related with the ground truth. For this purpose, the Jaccard Index [3] is used, which is a measure for the overlap between two clusterings: if n11 is the number of pairs of items that are clustered together in clustering a and b, n10 is the number of pairs of items that are clustered together in a but not in b, and n01 vice versa, then the Jaccard Index of a and b is:

$$JI(a,b) = \frac{n11}{n11 + n10 + n01}$$

4.2 Results MUNICIP: 5 municipalities

The results in terms of F1-score are visualised in Figure 2. A number of observations can be made from this figure. First, observe that most F1-scores are rather low. This is due to the fact that dividing 2502 distinct process instances in 5 clusters still leads to rather large clusters. Therefore, the precision of the process models corresponding with these clusters is rather low. Secondly, the horizontal line on Figure 2 represents the F-score of the unclustered event log, which is just around 0.39. Given the results of the other techniques, we see that a surprising number of trace clustering techniques do not improve compared to an unclustered event log. Most notable are the process-aware techniques *ActFreq* and *ActMRA*, who optimize for fitness over precision. Of all techniques, the only technique to significantly outperform the others is the constrained process model aware technique, *ConDriTraC*. When given the lowest percentage of constraints, it attains a weighted average F1-score of 0.63, the configurations with the 5 and 10% constraint set reach 0.48 and 0.49 respectively. In terms of relative improvement, the constrained direct (*ConGED*) and vector-based (*ConMRA, Con3-gram*) all score worse than the best pure model-driven technique (*ActMRA*), with RI < 1. The process model aware techniques, *ConDriTraC-1%, ConDriTraC-5%*, and *ConDriTraC-10%*, attain relative improvements of 1.56, 1.18, and 1.21, respectively.

Next, Figures 4 and 5 present the percentage of violated must-link and cannot-link constraints respectively, for each of the techniques and each of the percentage of constraints.

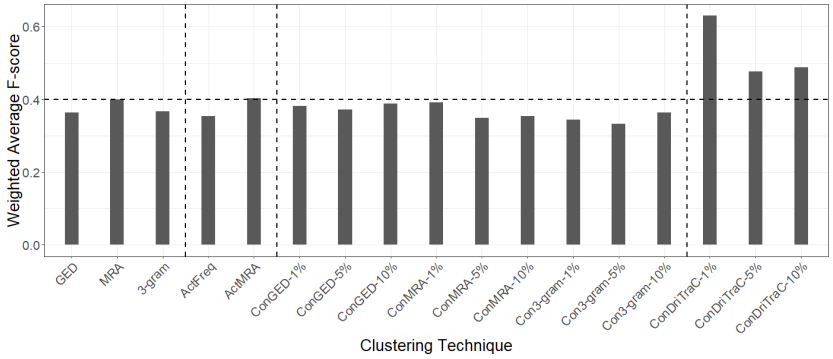

Fig. 2 Weighted Average F1-score results on MUNICIP. The horizontal dashed line indicates the baseline F-score without applying any clustering to the data.

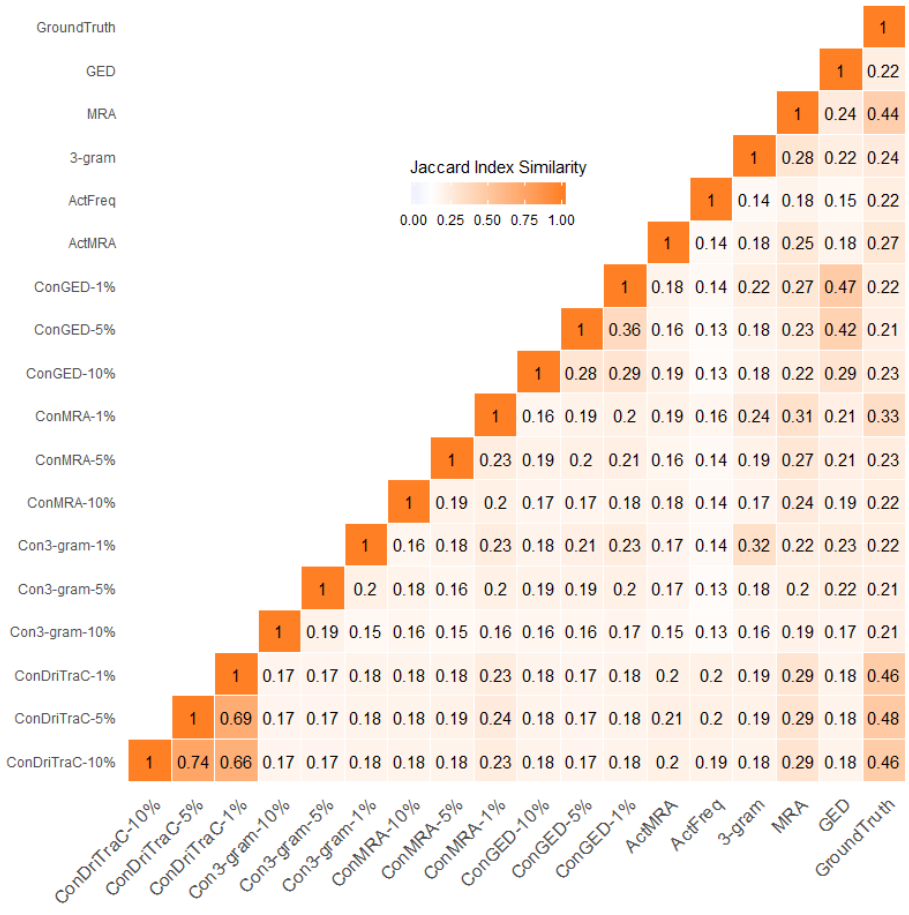

Fig. 3 HeatMap representing the Pairwise similarity of the clustering results of each of the clustering techniques, measured using the Jaccard Index on MUNICIP

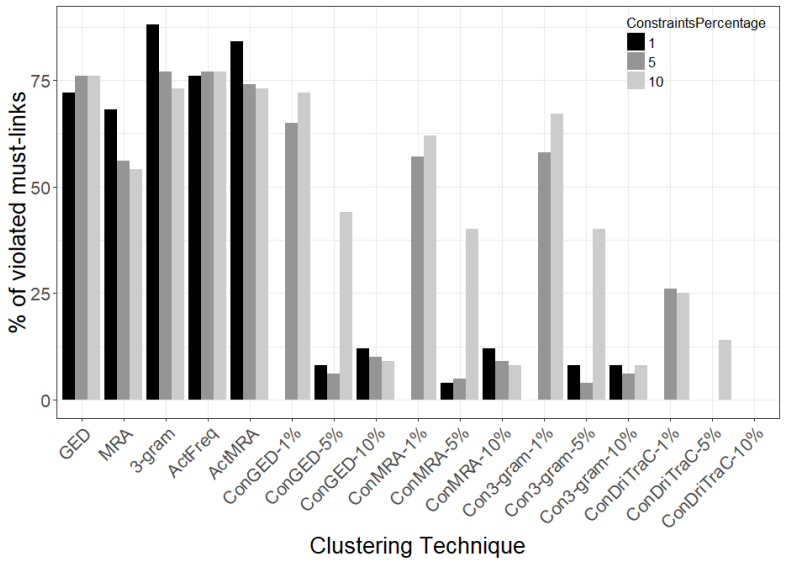

Fig. 4 Average percentage of violated must-link constraints for MUNICIP

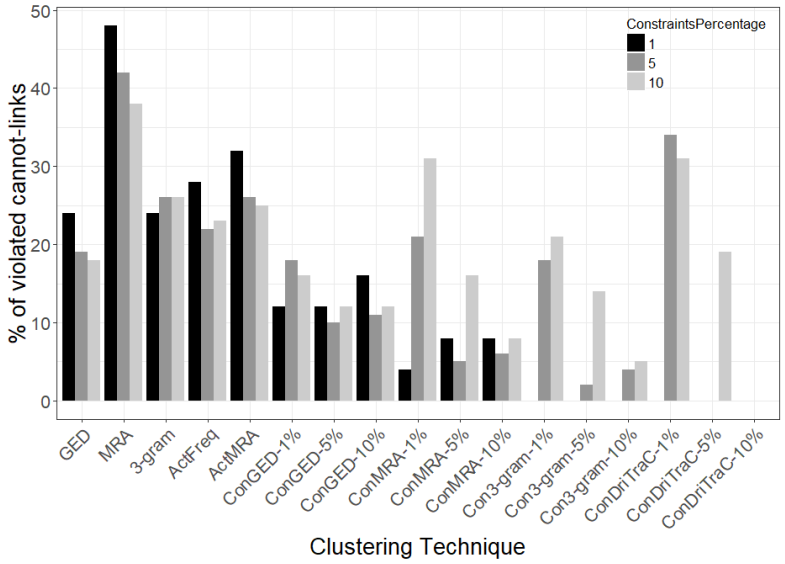

Fig. 5 Average percentage of violated cannot-link constraints for MUNICIP

A couple of remarks can be made: first, observe that *ConDriTraC-10%* is the only technique that doesn't violate any of the constraint sets. *ConDriTraC-1%and ConDriTraC-5%*, and the other constrained clustering techniques, violate a fraction of the constraints they were not provided with. The non-process model

aware techniques, *ConGED, ConMRA* and *Con3-gram*, also violate some of the constraints they were given, with the exception of *Con3-gram-1%*.

Finally, Figure 3 represents the similarity of the clustered event logs obtained by each of the clustering techniques. Most interesting is the similarity of the results to the ground truth. Here, we see that the process model aware expert-driven trace clustering technique *ConDriTraC* performs best, closely followed by the unconstrained *MRA*. Of the constrained techniques that are not model aware, *ConMRA* performs best, though its results are closer to the ground truth with less constraints. Finally, observe the high similarity between *GED* and *ConGED*, which decreases when more constraints are added, in line with expectations. Furthermore, the high similarity among the results of *ConDriTraC* based on different constraint sets is noticeable.

4.3 Results TABREAD: tablet newspaper reading

The results in terms of F1-score are visualised in Figure 6. Similar to the results on MUNICIP, the F1-scores remain rather low. The reason is still similar: dividing 2794 distinct process instances in 4 to 8 clusters still leads to rather large clusters. Therefore, the precision of the process models corresponding with these clusters is still rather low. Nonetheless, there is a clear improvement when clustering the log on this dataset: compared to the unclustered event log, represented by the horizontal line (F-score of 0.105), all settings lead to increased F1 scores. Secondly, observe that in general, more clusters leads to higher scores, in line with expectations. For the non expert-driven trace clustering techniques (*GED, MRA, 3-gram, ActFreq* and *ActMRA*), the latter two outperform the former three, in line with the findings of [14]. Finally, for the expert-driven techniques *ConDriTraC-5%* performs best for 8 clusters, with *ConDriTraC-10%* performing best for 6 clusters. For 4 clusters, the best result is obtained by *ActMRA*.

Table 4 Relative improvement of the constrained process model aware clustering solutions compared to the unconstrained process model aware solutions on the TABREAD dataset

	4 clusters	6 clusters	8 clusters
RI(*ConDriTraC-1%*, *ActFreq*)	1.10	1.03	0.82
RI(*ConDriTraC-1%*, *ActMRA*)	0.95	1.13	0.92
RI(*ConDriTraC-5%*, *ActFreq*)	0.75	0.99	1.15
RI(*ConDriTraC-5%*, *ActMRA*)	0.65	1.09	1.29
RI(*ConDriTraC-10%*, *ActFreq*)	0.80	1.19	0.90
RI(*ConDriTraC-10%*, *ActMRA*)	0.69	1.31	1.02

For a closer comparison of the results of the expert-driven techniques, Table 4 contains the relative improvement of the constrained process model aware techniques compared to the unconstrained process model aware trace clustering technique (*ActFreq* and *ActMRA*). For 4 clusters, only *ConDriTraC-1%* outperforms the non-constrained techniques. For 6 clusters, all constrained solutions perform better or on par with the unconstrained process model aware techniques. Finally, for

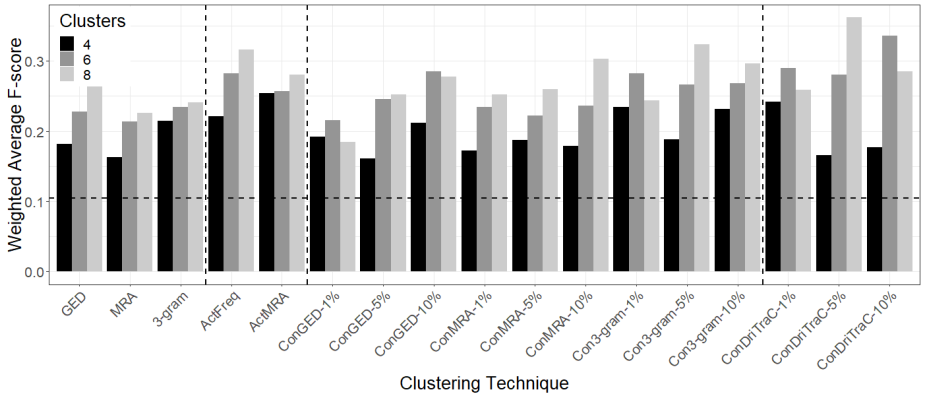

Fig. 6 Weighted Average F1-score results for different clustering techniques and number of clusters on TABREAD

8 clusters, the results are split evenly over the unconstrained and constrained solutions.

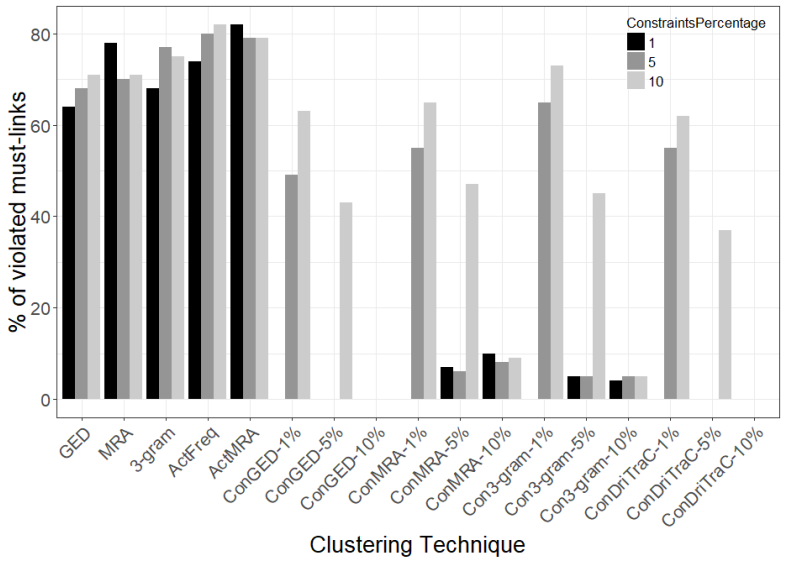

Fig. 7 Percentage of violated must-link constraints averaged across 4, 6 and 8 clusters on TABREAD

Next, Figures 7 and 8 present the percentage of violated must-link and cannot-link constraints respectively, for each of the techniques and each percentage level of constraints. The results are averaged across 4, 6 and 8 clusters. These results confirm the findings from the previous section regarding MUNICIP. Observe that the must-link constraints have a higher informativeness than the cannot-link

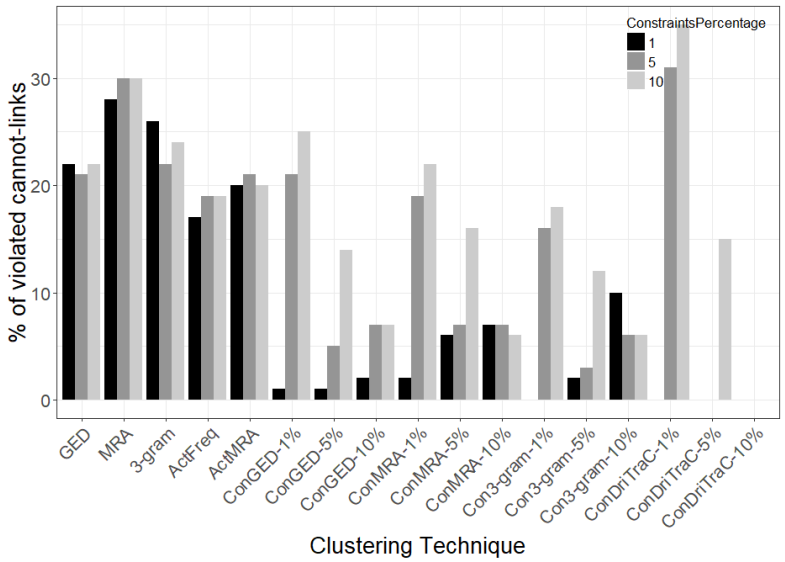

Fig. 8 Percentage of violated cannot-link constraints averaged across 4, 6 and 8 clusters on TABREAD

constraints: a higher proportion of must-link constraints is violated by unconstrained techniques compared to cannot-link constraints. This makes sense, especially for higher numbers of clusters: if there are 8 clusters to which one can assign traces, it is less likely that two traces would be assigned to the same cluster by a non-constrained clustering technique, making cannot-link violations less likely, and must-link violations more likely. This is represented in the data: most unconstrained clustering solutions violate between 50 and 80% of all must-link constraints, whereas they tend to violate 15 to 30% of cannot-link constraints.

Finally, the extent to which the results found are in line with the expectations of the expert (the ground truth) is visualized in Figure 9. It is clear that none of the techniques approach the expert's knowledge exceptionally well, with *MRA* and *ConDriTraC-1%* performing best with a similarity to the ground truth of 0.22. This is especially noticeable when compared to the highest similarity to ground truth on MUNICIP (the previous dataset), where a 0.48 similarity was reported for *ConDriTraC-5%*. This leads us to believe that on MUNICIP, there is some merit to the ground truth (the 5 distinct municipalities), whereas on TABREAD, the ground truth (a customer profiling provided by a marketeer) is less in line with a trace clustering view of the data.

4.4 Discussion

The experimental evaluations of each of the two datasets show that overall, our techniques are indeed capable of discovering trace clustering solutions that are in line with expert input (i.e. not violating constraints). Furthermore, we show that the constrained clustering techniques can maintain a comparable clustering quality, as

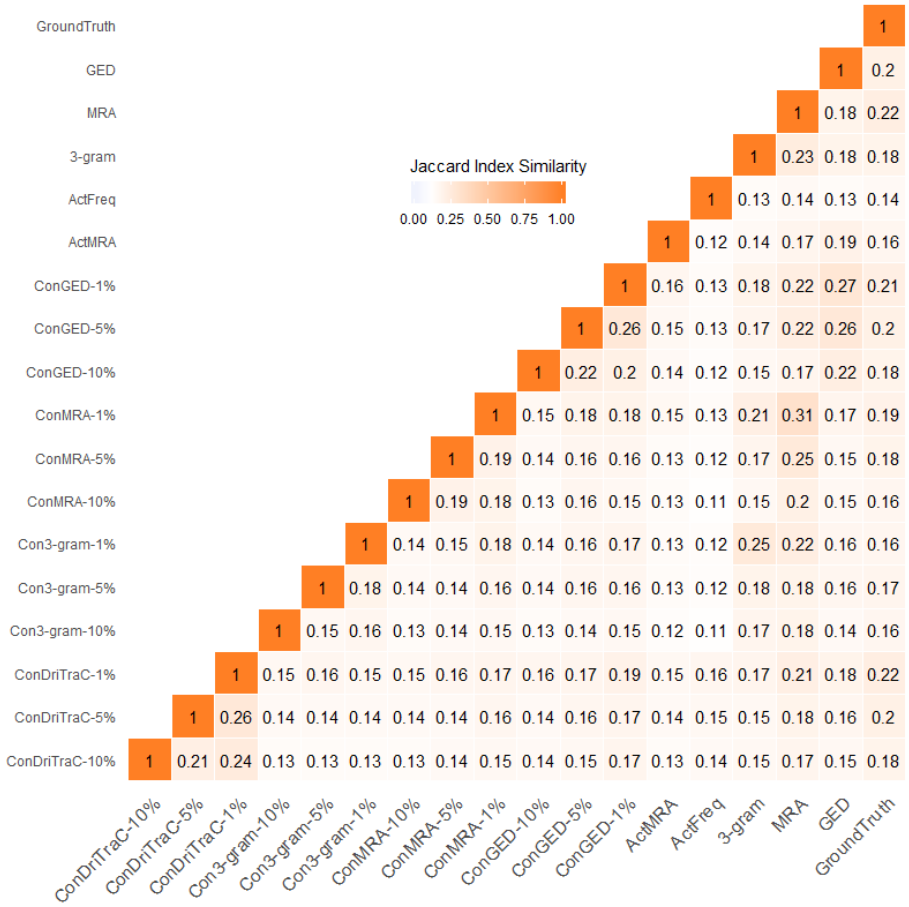

Fig. 9 HeatMap representing the Pairwise similarity of the clustering results of each of the clustering techniques, measured using the Jaccard Index, averaged over the cluster numbers, on TABREAD

measured by an aggregated F1-score of all underlying process models of a clustering. For the MUNICIP dataset and for particular settings of the ConDriTrac algorithm for the TABREAD dataset, the inclusion of expert knowledge even improves the clustering quality. This is due to the fact that trace clustering techniques in general cannot guarantee optimality due to computational complexity of the problem. As for limitations, it is important to point out that although the findings for both datasets are congruent, generalization towards other datasets is not proven. Moreover, our analysis shows that making a comparison with a ground truth is challenging. On the one hand, for MUNICIP, the trace clustering techniques might be picking up on some other information in the traces rather than whether they were executed within a particular municipality. For TABREAD, the ground truth was based on an extrapolation of reading profiles as defined by an expert. In the latter case, these reading profiles corresponded less to a process-based view on newspaper reading.

5 Conclusion

In a situation where an expert has a preconceived notion of what a clustering should look like, it is unlikely that a trace clustering algorithm will lead to clusters which are in line with his or her expectations. This paper proposes *expert-driven trace clustering* techniques that balance improvement in terms of trace clustering quality with the challenge of making clusters more justifiable for the expert. In an experimental evaluation, we have shown that existing trace clustering techniques lead to solutions that will violate the expert's expectations. Our proposed constrained trace clustering techniques successfully combine the strength of existing trace clustering techniques and constrained clustering approaches. Furthermore, this papers presents multiple benefits for practitioners: first, all proposed techniques are publicly available. Secondly, we have shown, both intuitively and in our experiments, how must-link constraints are more powerful than cannot-link constraints. This entails that it is more useful to invest time into extracting must-link constraints than cannot-link constraints in a practical setting. Finally, our approach for evaluating trace clustering solutions can be a guideline for deciding which solution to choose, whether it is based on process model quality, or by evaluating the usefulness of the constraint set.

Nonetheless, interesting avenues for future research remain. First, our approach could be applied to more data sets, both real life and artificial. Secondly, the usefulness of other types of constraints, such as constraints based on case-specific data, is a topic for further investigation. Finally, active constraint selection, in which the user is asked to provide answers to constraint queries during the clustering process, could lead to intriguing insights as well.

Acknowledgements

This research has been financed in part by the EC H2020 MSCA RISE NeEDS Project (Grant agreement ID: 822214)

References

1. Van der Aalst W, Adriansyah A, van Dongen B (2012) Replaying history on process models for conformance checking and performance analysis. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 2(2):182–192
2. Augusto A, Conforti R, Dumas M, La Rosa M, Polyvyanyy A (2018) Split miner: automated discovery of accurate and simple business process models from event logs. Knowledge and Information Systems doi:10.1007/s10115-018-1214-x
3. Ben-Hur A, Elisseeff A, Guyon I (2001) A stability based method for discovering structure in clustered data. In: Pacific symposium on biocomputing, vol 7, pp 6–17
4. Bose RPJC, van der Aalst WMP (2009) Context Aware Trace Clustering: Towards Improving Process Mining Results. Sdm pp 401–412, doi:10.1137/1.9781611972795.35

5. Bose RPJC, van der Aalst WMP (2010) Trace clustering based on conserved patterns: Towards achieving better process models. In: Lect. Notes Bus. Inf. Process., vol 43 LNBIP, pp 170–181, doi:10.1007/978-3-642-12186-9_16
6. Chen J, Huang X, Kanj IA, Xia G (2006) Strong computational lower bounds via parameterized complexity. Journal of Computer and System Sciences 72(8):1346–1367
7. Davidson I, Ravi SS (2005) Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 3721 LNAI:59–70, doi:10.1007/11564126_11
8. Davidson I, Wagstaff KL, Basu S (2006) Measuring Constraint-Set Utility for Partitional Clustering Algorithms. Proc 10th Eur Conf Princ Pract Knowl Discov Databases pp 115–126, doi:10.1007/11871637_15
9. De Koninck P, De Weerdt J, vanden Broucke SKLM (2017) Explaining clusterings of process instances. Data Mining and Knowledge Discovery 31(3):774–808, doi:10.1007/s10618-016-0488-4
10. De Koninck P, Nelissen K, Baesens B, vanden Broucke S, Snoeck M, De Weerdt J (2017) An approach for incorporating expert knowledge in trace clustering. In: Dubois E, Pohl K (eds) Advanced Information Systems Engineering: 29th International Conference, CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings, Springer International Publishing, Cham, pp 561–576, doi:10.1007/978-3-319-59536-8_35
11. De Smedt J, De Weerdt J, Vanthienen J, Poels G (2016) Mixed-paradigm process modeling with intertwined state spaces. Business & Information Systems Engineering 58(1):19–29, doi:10.1007/s12599-015-0416-y
12. De Weerdt J, De Backer M, Vanthienen J, Baesens B (2011) A robust f-measure for evaluating discovered process models. In: Computational Intelligence and Data Mining (CIDM), 2011 IEEE Symposium on, IEEE, pp 148–155, doi:10.1109/CIDM.2011.5949428
13. De Weerdt J, De Backer M, Vanthienen J, Baesens B (2012) A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs. Inf Syst 37(7):654–676, doi:10.1016/j.is.2012.02.004
14. De Weerdt J, vanden Broucke S, Vanthienen J, Baesens B (2013) Active trace clustering for improved process discovery. IEEE Trans Knowl Data Eng 25(12):2708–2720, doi:10.1109/TKDE.2013.64
15. Delias P, Doumpos M, Grigoroudis E, Manolitzas P, Matsatsinis N (2015) Supporting healthcare management decisions via robust clustering of event logs. Knowledge-Based Syst 84:203–213, doi:10.1016/j.knosys.2015.04.012
16. Dumas M, Rosa ML, Mendling J, Reijers HA (2018) Fundamentals of Business Process Management, Second Edition. Springer, doi:10.1007/978-3-662-56509-4, URL `https://doi.org/10.1007/978-3-662-56509-4`
17. Eaton E, desJardins M, Jacob S (2014) Multi-view constrained clustering with an incomplete mapping between views. Knowledge and Information Systems 38(1):231–257, doi:10.1007/s10115-012-0577-7
18. Goedertier S, Martens D, Vanthienen J, Baesens B (2009) Robust Process Discovery with Artificial Negative Events. J Mach Learn Res 10:1305—-1340
19. Klein D, Kamvar SD, Manning CD (2002) From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. Tech. rep., Stanford

20. Law M, Topchy A, Jain A (2005) Model-based Clustering With Probabilistic Constraints. Sdm pp 1–5, doi:10.1137/1.9781611972757.77
21. Leemans SJJ, Fahland D, van der Aalst WMP (2013) Discovering Block-Structured Process Models from Event Logs - A Constructive Approach. In: Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), Springer, pp 311–329, doi:10.1007/978-3-642-38697-8_17
22. Mabroukeh NR, Ezeife CI (2010) A taxonomy of sequential pattern mining algorithms. ACM Comput Surv 43(1):3:1–3:41, doi:10.1145/1824795.1824798
23. Mannhardt F, de Leoni M, Reijers HA, van der Aalst WM, Toussaint PJ (2016) From low-level events to activities - a pattern-based approach. In: 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, Springer, LNCS, pp 125–141, doi:10.1007/978-3-319-45348-4_8
24. Martens D, Vanthienen J, Verbeke W, Baesens B (2011) Performance of classification models from a user perspective. Decision Support Systems 51(4):782 – 793, doi:10.1016/j.dss.2011.01.013
25. Muñoz-Gama J, Carmona J (2010) A fresh look at precision in process conformance. In: Hull R, Mendling J, Tai S (eds) Business Process Management: 8th International Conference, BPM 2010, Hoboken, NJ, USA, September 13-16, 2010. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 211–226, doi:10.1007/978-3-642-15618-2_16
26. Murtagh F (1984) A survey of recent advances in hierarchical clustering algorithms which use cluster centers. The Computing Journal 26:354–359
27. Rozinat A, Van der Aalst WM (2008) Conformance checking of processes based on monitoring real behavior. Information Systems 33(1):64–95
28. Song M, Günther C, van der Aalst WMP (2009) Trace Clustering in Business Process Mining. In: Bus. Process Manag. Work., Springer, vol 17, pp 109–120, doi:10.1007/978-3-642-00328-8_11
29. Tax N, Sidorova N, Haakma R, van der Aalst WMP (2016) Mining local process models. Journal of Innovation in Digital Ecosystems 3(2):183–196, doi:10.1016/j.jides.2016.11.001
30. van der Aalst WMP, Adriansyah A, Van Dongen B (2012) Replaying history on process models for conformance checking and performance analysis. Wiley Interdiscip Rev Data Min Knowl Discov 2(2):182–192, doi:10.1002/widm.1045
31. Van Dongen B (2015) Bpi challenge 2015 (dataset). doi:10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1
32. vanden Broucke S, De Weerdt J, Vanthienen J, Baesens B (2014) Determining process model precision and generalization with weighted artificial negative events. IEEE Trans Knowl Data Eng 26(8):1877–1889
33. vanden Broucke SKLM, De Weerdt J (2017) Fodina: A robust and flexible heuristic process discovery technique. Decision Support Systems 100(Supplement C):109 – 118, doi:https://doi.org/10.1016/j.dss.2017.04.005, smart Business Process Management
34. Veiga GM, Ferreira DR (2010) Understanding spaghetti models with sequence clustering for prom. In: Rinderle-Ma S, Sadiq S, Leymann F (eds) Business Process Management Workshops, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 92–103
35. Wagstaff K, Cardie C, Rogers S, Schroedl S (2001) Constrained k-means clustering with background knowledge. In: In ICML, Morgan Kaufmann, pp 577–584

36. Wang N, Sun S, OuYang D (2016) Business process modeling abstraction based on semi-supervised clustering analysis. Business & Information Systems Engineering doi:10.1007/s12599-016-0457-x
37. Wang X, Davidson I (2010) Flexible constrained spectral clustering. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, KDD '10, pp 563–572, doi:10.1145/1835804.1835877
38. Weijters A, van der Aalst WMP, De Medeiros AA (2006) Process mining with the heuristics miner-algorithm. Technische Universiteit Eindhoven, Tech Rep WP 166:1–34
39. Zhu S, Wang D, Li T (2010) Data clustering with size constraints. Knowledge-Based Systems 23(8):883 – 889, doi:http://dx.doi.org/10.1016/j.knosys.2010.06.003